



# PRÉVISION DE LA VARIATION DU CHÔMAGE AU MAROC

proposé par:  
YASSINE ALAMRANI

préparé par  
OMAIMA YOUNES  
AHLAM EL GAMANI  
AYA EL HAOUDAR  
HAMZA KHOLTI  
ANOUAR BOUZHAR

# PLAN DU PRESENTATION :

- Collecte de données.
- prétraitement.
- visualisation des données.
- Interprétation.
- Choix des modèles.
- Construction des modèles.
- validation et utilisation des modèles



# INTRODUCTION :

Le taux de chômage, reflet de la vitalité économique d'un pays, joue un rôle crucial dans la vie de ses citoyens. Cette brève exploration examinera les récents développements de ce paramètre au Maroc, mettant en lumière les forces motrices derrière ces changements. Plongeons ensemble dans l'analyse de cette dynamique économique essentielle et envisageons les tendances futures qui façonneront le paysage de l'emploi au Maroc.



# 1. RESSOURCES DE DONNÉES:

The screenshot shows the homepage of the High Commission for Planning (HCP) in Morocco. The top navigation bar includes links for "TOUT SUR HCP", "VIDEOTHEQUE", "MEDIA", and "CONTACTS". The main header features the HCP logo and Arabic text: "المملكة المغربية" (Kingdom of Morocco), "المندوبية السامية للتخطيط" (High Commission for Planning), and "HAUT-COMMISSARIAT AU PLAN". A search bar is also present. Below the header, there are five main menu categories: "STATISTIQUES & ÉTUDES", "BASES DE DONNÉES", "VISUALISATION", "PUBLICATIONS", and "MÉTHODOLOGIES". The "VISUALISATION" section is currently selected and displays a chart titled "Taux de chômage annuel national selon le diplôme" (Annual national unemployment rate by education level). The chart compares four groups: "Sans diplôme" (blue line), "Ayant un diplôme: Niveau moyen" (red line), "Ayant un diplôme: Niveau supérieur" (yellow line), and "Ensemble" (green line). The yellow line (higher education) consistently shows the highest unemployment rates, peaking at approximately 27% in 2023. The red line (medium education) shows the lowest rates, around 21%. The blue line (no diploma) and green line (overall average) fall in between. The left sidebar has a "Marché du travail" menu with "Activité", "Emploi", and "Chômage" options, and an "Accès rapide" section with various icons. The bottom navigation bar includes links for weather (13°C), search, and various software applications like Google Chrome, Microsoft Word, and WhatsApp.

TOUT SUR HCP VIDEOTHEQUE MEDIA CONTACTS

الملكية المغربية  
المندوبية السامية للتخطيط  
+٢٠٣٤٥٤٤ | +٢٠٣٤٦٥٥

HAUT-COMMISSARIAT AU PLAN

Recherche

STATISTIQUES & ÉTUDES BASES DE DONNÉES VISUALISATION PUBLICATIONS MÉTHODOLOGIES

Chômage

Marché du travail

Activité

Emploi

Chômage

Accès rapide

13°

Accueil > Marché du travail > Chômage > Indicateurs Chômage

Taux de chômage annuel national selon le diplôme

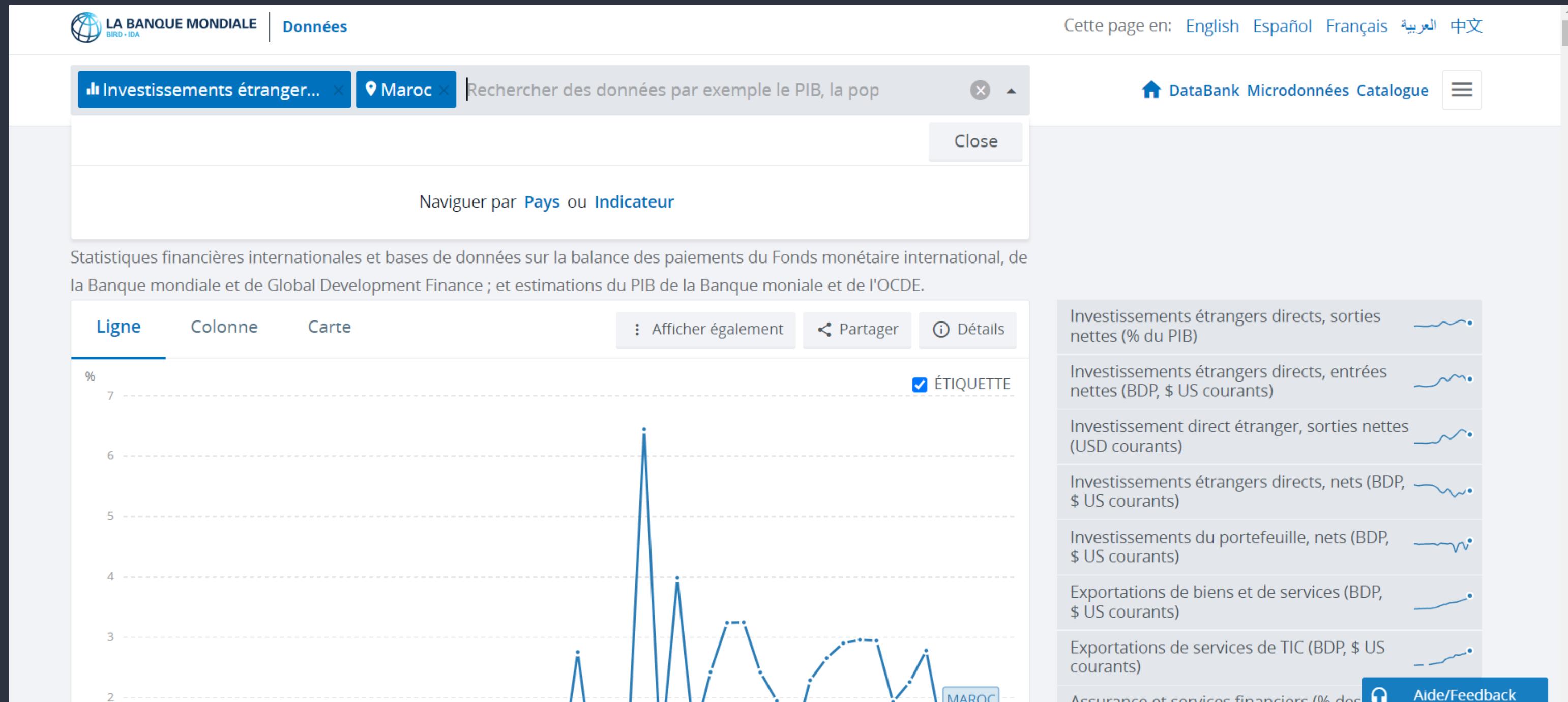
Sans diplôme Ayant un diplôme: Niveau moyen Ayant un diplôme: Niveau supérieur Ensemble

30,0  
25,0  
20,0

25,8

FRA FR 22:56 2023-12-31

# 1. RESSOURCES DE DONNÉES:



# 1.COLLECTE DE DONNÉES:

Le Taux De Chômage Annuel Selon Les Tranches D'âge  
Population Masculine

A	B	C	D	E	F
Années	15 - 24	25 - 34	35 - 44	45 et plus	Ensemble
2022	28.7	16.4	5.6	3.3	10.3
2021	28.4	17.2	6.2	3.7	10.9
2020	28.0	16.2	6.3	3.9	10.7
2019	22.0	12.6	3.8	2.0	7.8
2018	22.5	12.4	3.6	2.1	8.1
2017	23.8	12.7	4.2	2.4	8.8
2016	22.0	11.9	4.3	2.9	8.9
2015	20.6	12.9	5.2	3.4	9.4
2014	20.3	12.7	5.8	3.4	9.7
2013	19.5	12.3	5.3	2.6	9.1
2012	18.4	12.2	4.6	2.0	8.7
2011	18.1	11.5	4.5	1.9	8.4
2010	18.1	11.9	4.9	2.3	8.9
2009	18.5	11.7	5.2	2.1	9.0
2008	19.1	12.8	4.8	2.3	9.5
2007	17.9	13.8	5.6	2.3	9.8
2006	17.5	13.2	5.7	2.6	9.7
2005	17.8	15.8	6.6	1.9	11.0
2004	17.4	15.6	6.8	2.5	10.7
2003	16.8	16.3	6.8	2.8	11.2
2002	17.4	16.2	6.0	2.3	11.1
2001	19.3	18.1	6.5	2.4	12.3
2000	21.1	19.8	7.5	3.0	13.6
1999	21.9	20.2	7.6	3.3	14.1

Le Taux De Chômage Annuel Selon Les Tranches D'âge  
Population Féminine

A	B	C	D	E	F
Années	15 - 24	25 - 34	35 - 44	45 et plus	Ensemble
2022	44.4	28.0	9.1	3.2	17.2
2021	41.9	26.9	9.7	4.0	16.8
2020	41.2	26.2	8.9	4.0	16.2
2019	33.4	22.9	6.6	2.3	13.5
2018	34.3	23.2	7.6	2.3	14.1
2017	34.3	23.8	8.0	2.5	14.7
2016	22.8	17.3	6.2	2.1	10.9
2015	21.4	16.8	6.3	2.0	10.5
2014	19.1	17.0	7.0	2.1	10.4
2013	18.1	15.6	6.4	1.5	9.6
2012	19.2	16.1	6.3	1.6	9.9
2011	17.4	16.9	7.0	1.5	10.2
2010	16.1	15.2	6.9	1.6	9.6
2009	16.2	15.5	5.9	1.4	9.5
2008	16.1	15.5	6.4	1.4	9.8
2007	15.5	15.8	6.5	1.2	9.8
2006	14.1	16.0	6.6	1.4	9.7
2005	14.9	19.4	6.1	0.5	11.3
2004	14.9	18.6	6.6	1.3	11.1
2003	15.1	21.2	7.5	1.4	12.4
2002	16.2	20.5	6.5	1.2	12.1
2001	16.2	21.2	6.1	1.6	12.2
2000	15.8	22.1	6.6	1.5	12.8
1999	16.6	21.8	6.7	1.7	13.2

# 1.COLLECTE DE DONNÉES:

Le Taux de Chômage Annuel Selon Le Diplôme  
Population Masculine

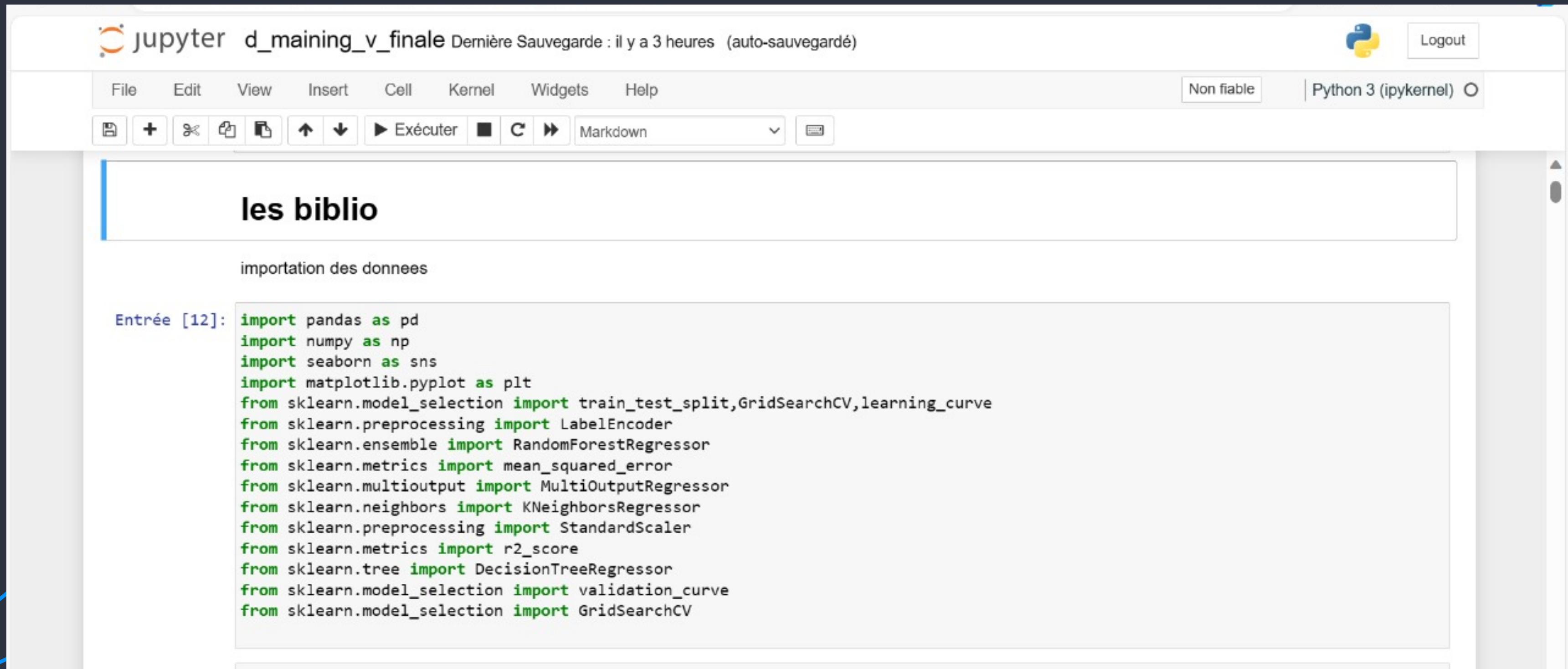
A	B	C	D	E	
Années	Sans diplôme	Ayant un diplôme: Niveau moyen	Ayant un diplôme: Niveau supérieur	Ensemble	
2022	4.4	13.0	20.8	10.3	
2021	4.6	14.5	21.9	10.9	
2020	5.8	14.0	19.6	10.7	
2019	3.2	10.8	17.1	7.8	
2018	3.4	11.7	17.4	8.1	
2017	3.9	13.0	17.9	8.8	
2016	4.1	13.1	17.8	8.9	
2015	4.6	14.4	17.9	9.4	
2014	5.4	14.3	17.1	9.7	
2013	5.2	14.2	14.8	9.1	
2012	4.6	14.3	14.0	8.7	
2011	4.5	14.0	14.4	8.4	
2010	5.0	14.8	14.3	8.9	
2009	5.1	15.5	14.2	9.0	
2008	5.5	16.8	14.9	9.5	
2007	5.6	16.9	16.8	9.8	
2006	5.6	17.3	15.7	9.7	
2005	6.1	19.1	22.0	11.0	
2004	5.8	19.4	22.3	10.7	
2003	6.1	20.3	22.3	11.2	

Le Taux De Chômage Annuel selon Le Diplôme  
Population Féminine

A	B	C	D	E	
Années	Sans diplôme	Ayant un diplôme: Niveau moyen	Ayant un diplôme: Niveau supérieur	Ensemble	
2022	3.8	21.5	34.8	17.2	
2021	4.5	24.5	32.8	16.8	
2020	4.8	24.6	31.8	16.2	
2019	2.9	21.3	29.5	13.5	
2018	3.2	23.2	32.5	14.1	
2017	3.7	25.8	33.0	14.7	
2016	2.8	19.1	29.8	10.9	
2015	3.0	21.1	27.3	10.5	
2014	2.9	21.8	28.3	10.4	
2013	2.6	20.0	26.7	9.6	
2012	2.8	21.0	27.4	9.9	
2011	2.8	22.8	28.5	10.2	
2010	3.2	22.1	25.3	9.6	
2009	2.8	22.8	26.0	9.5	
2008	2.9	23.6	28.2	9.8	
2007	3.1	23.6	28.5	9.8	
2006	3.1	25.2	27.1	9.7	
2005	3.1	29.4	35.6	11.3	
2004	3.3	29.0	35.1	11.1	
2003	4.0	30.5	36.8	12.4	
2002	3.8	29.2	35.3	12.1	
2001	4.1	30.1	35.5	12.2	

# 2. PRÉTRAITEMENT DE DONNÉES :

L'importation des bibliothèques



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter d\_maining\_v\_finale Dernière Sauvegarde : il y a 3 heures (auto-sauvegardé)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Non fiable, Python 3 (ipykernel)
- Buttons:** New, Open, Save, Cell, Run, Stop, Kernel, Help, Markdown.
- Section Header:** les biblio
- Text:** importation des données
- Code Cell [12]:**

```
Entrée [12]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,GridSearchCV,learning_curve
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.multioutput import MultiOutputRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import validation_curve
from sklearn.model_selection import GridSearchCV
```

## 2. PRÉTRAITEMENT DE DONNÉES :

Importation de Dataset:

```
Entrée [13]: data_age_feminin = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\age F.xlsx")
              data_age_masculin = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\age mascul.xlsx")
              data_dip_feminin = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\DIPLOME F.xlsx")
              data_dip_masculin = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\diplome M.xlsx")
```

Ajout de la colonne de sexe :

```
Entrée [14]: data_age_feminin['sex']='F'
              data_age_masculin['sex']='M'
              data_dip_feminin['sex']='F'
              data_dip_masculin['sex']='M'
```

```
Entrée [4]: data_dip_feminin.head()
```

	Années	Sans diplôme	Ayant un diplôme: Niveau moyen	Ayant un diplôme: Niveau supérieur	Ensemble	sex
0	2022	3.8	21.5	34.8	17.2	F
1	2021	4.5	24.5	32.8	16.8	F
2	2020	4.8	24.6	31.8	16.2	F
3	2019	2.9	21.3	29.5	13.5	F
4	2018	3.2	23.2	32.5	14.1	F

## 2. PRÉTRAITEMENT DE DONNÉES :

Regroupement selon l'âge :

```
Entrée [16]: data_age = pd.concat([data_age_feminin, data_age_masculin], axis=0)
```

```
Entrée [17]: data_dip = pd.concat([data_dip_feminin, data_dip_masculin], axis=0)
```

Le nombre de lignes et de colonnes :

```
Entrée [18]: data_age.shape
```

```
Out[18]: (48, 7)
```

```
Entrée [19]: data_dip.shape
```

```
Out[19]: (48, 6)
```

Overview de data de l'âge avec le sex : féminin et masculin:

```
Entrée [20]: data_age.head()
```

```
Out[20]:
```

	Années	15 - 24	25 - 34	35 - 44	45 et plus	Ensemble	sex
0	2022	44.4	28.0	9.1	3.2	17.2	F
1	2021	41.9	26.9	9.7	4.0	16.8	F
2	2020	41.2	26.2	8.9	4.0	16.2	F
3	2019	33.4	22.9	6.6	2.3	13.5	F
4	2018	34.3	23.2	7.6	2.3	14.1	F

## 2.PRÉTRAITEMENT DE DONNÉES :

Overview de data de Diplome avec le sex : féminin et masculin

Entrée [21]: `data_dip.head()`

Out[21]:

	Années	Sans diplôme	Ayant un diplôme: Niveau moyen	Ayant un diplôme: Niveau supérieur	Ensemble	sex
0	2022	3.8	21.5	34.8	17.2	F
1	2021	4.5	24.5	32.8	16.8	F
2	2020	4.8	24.6	31.8	16.2	F
3	2019	2.9	21.3	29.5	13.5	F
4	2018	3.2	23.2	32.5	14.1	F

Renommer les colonnes:

Entrée [22]: `data_age.columns=['Annes','15-24','25-34','35-44','plus que 45','Ensemble','sex']`

Entrée [23]: `data_dip.columns=['Annes','sans d','d moyenne','d superieur','Ensemble','sex']`

Suppression de colonne ensembles :

Entrée [24]: `data_dip = data_dip.drop(columns = ['Ensemble'])`

Entrée [25]: `data_age = data_age.drop(columns = ['Ensemble'])`

Overview de data de Diplome avec le sex : féminin et masculin

## 2.PRÉTRAITEMENT DE DONNÉES :

Concaténation des deux datasets ; le diplôme avec l'âge :

```
Entrée [26]: data_final = pd.concat([data_dip, data_age.drop(['Annes', 'sex'], axis=1)], axis = 1)
```

Le triage du Databset :

```
Entrée [27]: data_final = data_final.sort_values(by=['Annes', 'sex'], ascending=[False, False], inplace = False)
```

Le nombre des lignes et des colonnes :

```
Entrée [28]: data_final.shape
```

```
Out[28]: (48, 9)
```

## 2. PRÉTRAITEMENT DE DONNÉES :

Les Valeurs manquantes :

```
Entrée [29]: data_final.isnull().sum()
```

```
Out[29]: Annes          0  
sans d           0  
d moyenne        0  
d supérieur       0  
sex              0  
15-24            0  
25-34            0  
35-44            0  
plus que 45       0  
dtype: int64
```

Les types des variables :

```
Entrée [30]: data_final.dtypes
```

```
Out[30]: Annes          int64  
sans d           float64  
d moyenne        float64  
d supérieur       float64  
sex              object  
15-24            float64  
25-34            float64  
35-44            float64  
plus que 45       float64  
dtype: object
```

## 2.PRÉTRAITEMENT DE DONNÉES :

Une brève description de nos données. :

Entrée [31]: `data_final.describe()`

Out[31]:

	Annes	sans d	d moyenne	d supérieur	15-24	25-34	35-44	plus que 45
count	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000
mean	2010.500000	4.414225	20.729897	24.918585	21.466854	17.083513	6.267704	2.265942
std	6.995439	1.446039	5.944504	7.418594	7.383010	4.272988	1.267009	0.814612
min	1999.000000	2.647612	10.800000	13.964118	14.121230	11.466960	3.600000	0.482458
25%	2004.750000	3.185581	14.750457	17.875000	16.767681	13.142145	5.627507	1.591111
50%	2010.500000	4.219795	21.045649	24.580020	18.800561	16.194735	6.332842	2.183733
75%	2016.250000	5.447532	24.525000	30.300000	22.125000	19.935794	6.794961	2.683076
max	2022.000000	9.131952	34.725906	39.172855	44.400000	28.000000	9.700000	4.000000

## 2. PRÉTRAITEMENT DE DONNÉES :

changer les valeurs de sex par des valeurs numériques :

```
Entrée [32]: data_final['sex'] = data_final['sex'].replace({'M': 1, 'F': -1})
```

La résultat obtenu :

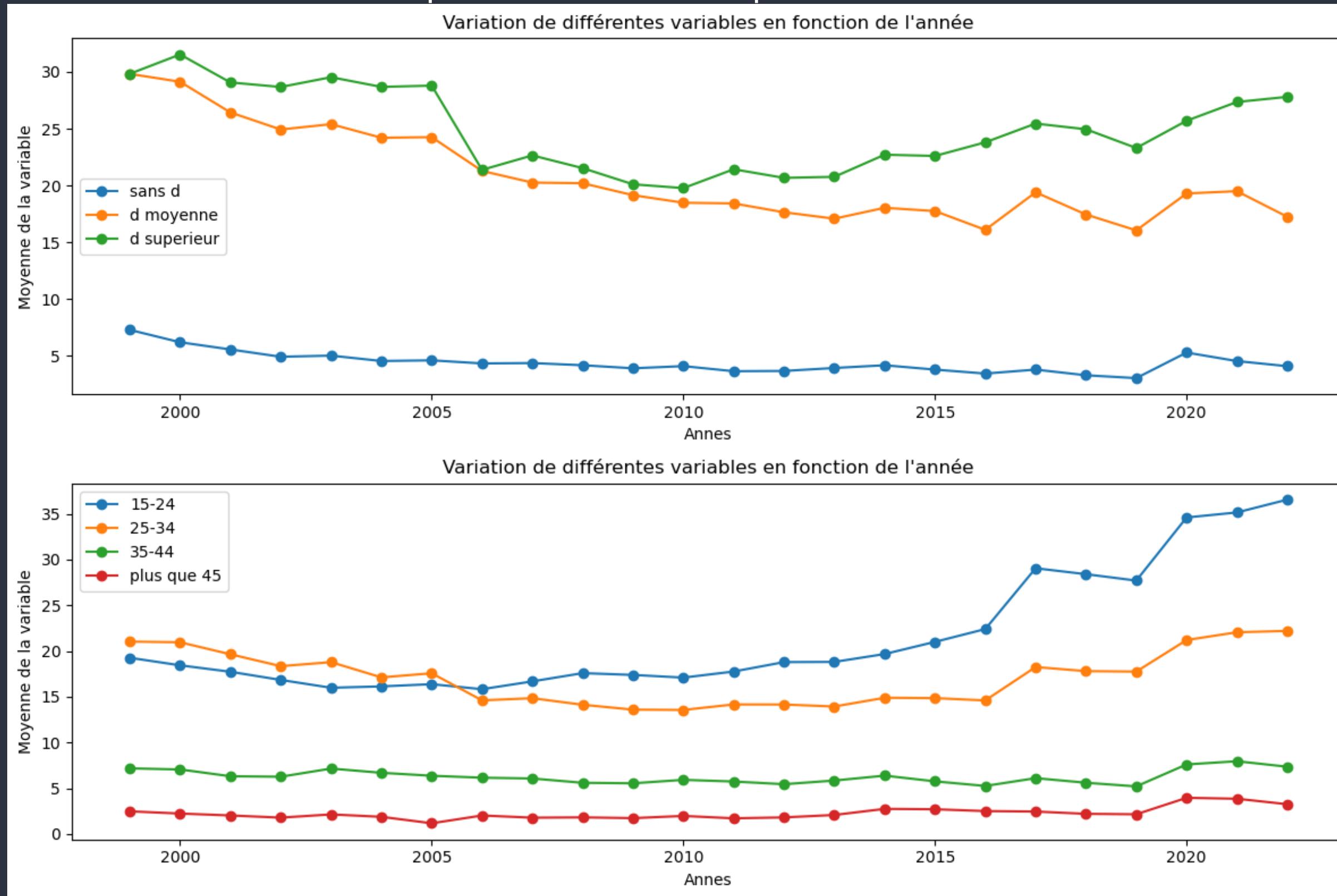
```
Entrée [33]: data_final.head()
```

Out[33]:

	Annes	sans d	d moyenne	d supérieur	sex	15-24	25-34	35-44	plus que 45
0	2022	4.4	13.0	20.8	1	28.7	16.4	5.6	3.3
0	2022	3.8	21.5	34.8	-1	44.4	28.0	9.1	3.2
1	2021	4.6	14.5	21.9	1	28.4	17.2	6.2	3.7
1	2021	4.5	24.5	32.8	-1	41.9	26.9	9.7	4.0
2	2020	5.8	14.0	19.6	1	28.0	16.2	6.3	3.9

# 3. VISUALISATION DES DONNÉES

Visualisation des Graphes De Variation de la moyenne de Diplôme et Des tranches d'âge au fil de temps sans tenir compte le sexe :



# 3. VISUALISATION DES DONNÉES

Importation de dataset des indicateurs investissement et Effet Sanitaire :

```
Entrée [155]: new_features = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\new_features.xlsx")
```

```
Entrée [156]: new_features.head()
```

Out[156]:

	Annes	effet sanitaire	investissement
0	2022	0.5	1.663588
1	2022	0.5	1.663588
2	2021	3.0	1.596519
3	2021	3.0	1.596519
4	2020	7.0	1.169073

# 3. VISUALISATION DES DONNÉES

Ordonner les indexes de DataFrame et concaténation du dataframe avec le data des indicateurs :

```
Entrée [141]: data_final = data_final.reset_index(drop=True)
```

```
Entrée [142]: data_final = pd.concat([data_final, new_features.drop(['Annes'], axis=1)], axis = 1)
```

```
Entrée [143]: data_final.head()
```

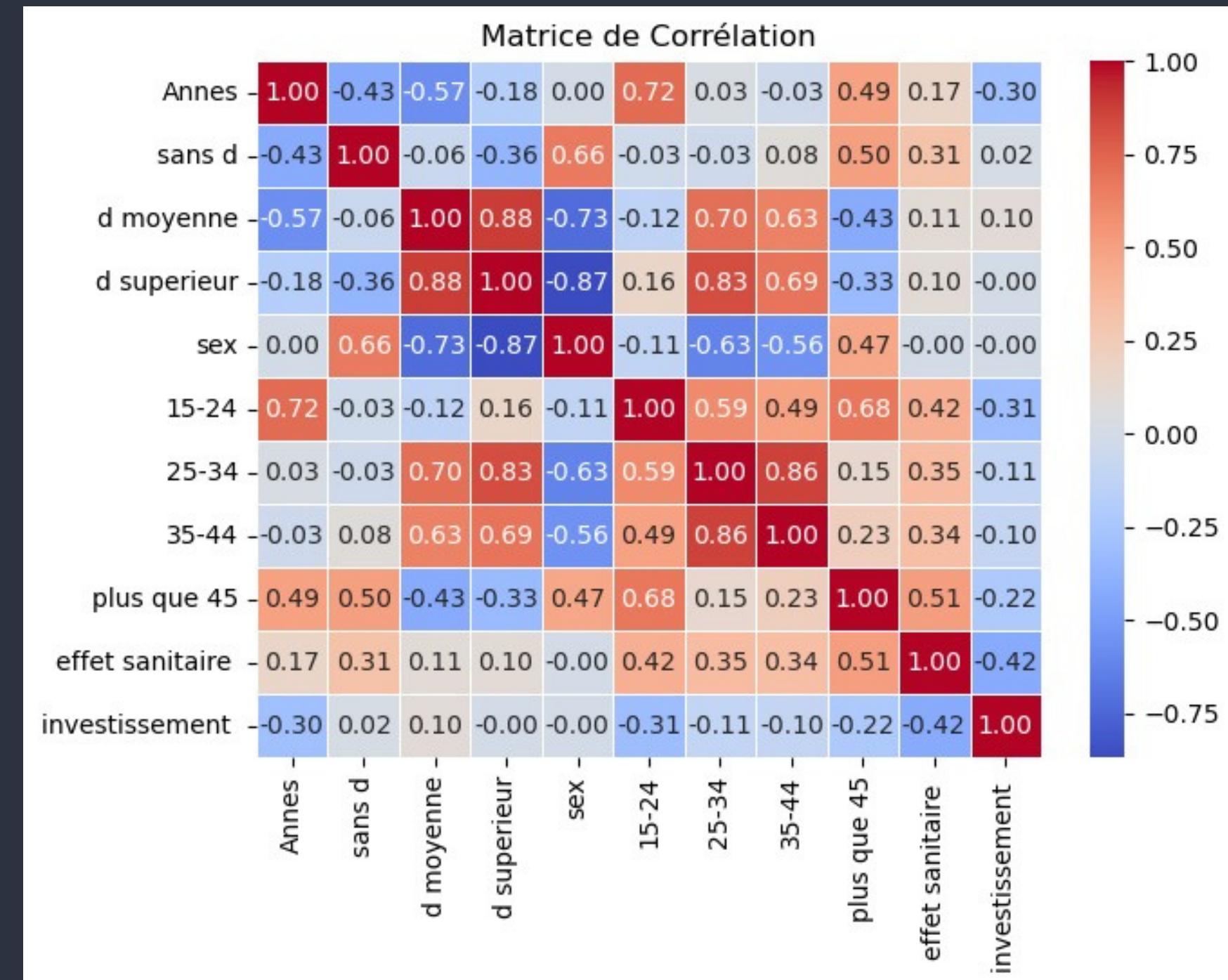
Affichage de DataFrame final :

Out[143]:

	Annes	sans d	d moyenne	d supérieur	sex	15-24	25-34	35-44	plus que 45	effet sanitaire	investissement	effet sanitaire	investissement	effet sanitaire	investissement
0	1.661325	4.4	13.0	20.8	1.0	28.7	16.4	5.6	3.3	-0.204390	-0.672202	0.5	1.663588	0.5	1.663588
1	1.661325	3.8	21.5	34.8	-1.0	44.4	28.0	9.1	3.2	-0.204390	-0.672202	0.5	1.663588	0.5	1.663588
2	1.516862	4.6	14.5	21.9	1.0	28.4	17.2	6.2	3.7	1.328538	-0.731341	3.0	1.596519	3.0	1.596519
3	1.516862	4.5	24.5	32.8	-1.0	41.9	26.9	9.7	4.0	1.328538	-0.731341	3.0	1.596519	3.0	1.596519
4	1.372399	5.8	14.0	19.6	1.0	28.0	16.2	6.3	3.9	3.781223	-1.108249	7.0	1.169073	7.0	1.169073

## Matrice de Corrélation qui montre l'Indépendance de différents variables entre eux :

```
Entrée [162]: correlation_matrix = data_final.corr()
plt.figure(figsize=(7, 5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Matrice de Corrélation')
plt.show()
```



# 3. VISUALISATION DES DONNÉES

Visualisation des graphes de l'effet sanitaire et l'investissement au fil de temps :

```
: features = ['investissement', 'effet sanitaire']
plt.figure(figsize=(20, 15))
for i in features:
    # Group by 'Annes' and calculate mean
    moyenne_par_annees = data_final.groupby('Annes')[i].mean()

    # Plot each line in the same plot
    plt.plot(moyenne_par_annees.index, moyenne_par_annees.values, marker='o', linestyle='--', label=i)

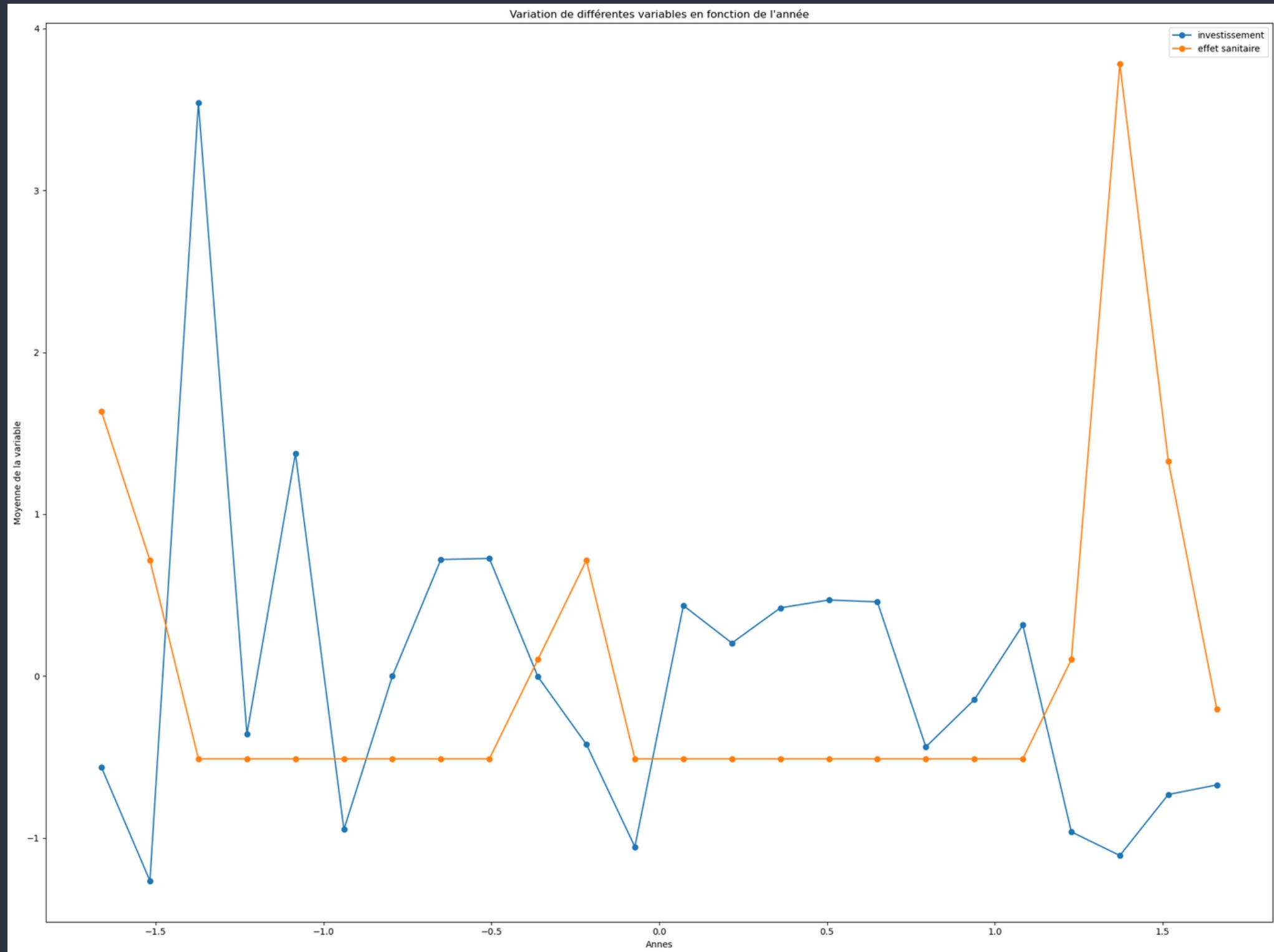
# Add Labels and title
plt.xlabel('Annes')
plt.ylabel('Moyenne de la variable')
plt.title("Variation de différentes variables en fonction de l'année")

# Add Legend
plt.legend()

plt.tight_layout()
plt.show()
```

# 3. VISUALISATION DES DONNÉES

La résultat obtenu de visualisation :



# 3. VISUALISATION DES DONNÉES

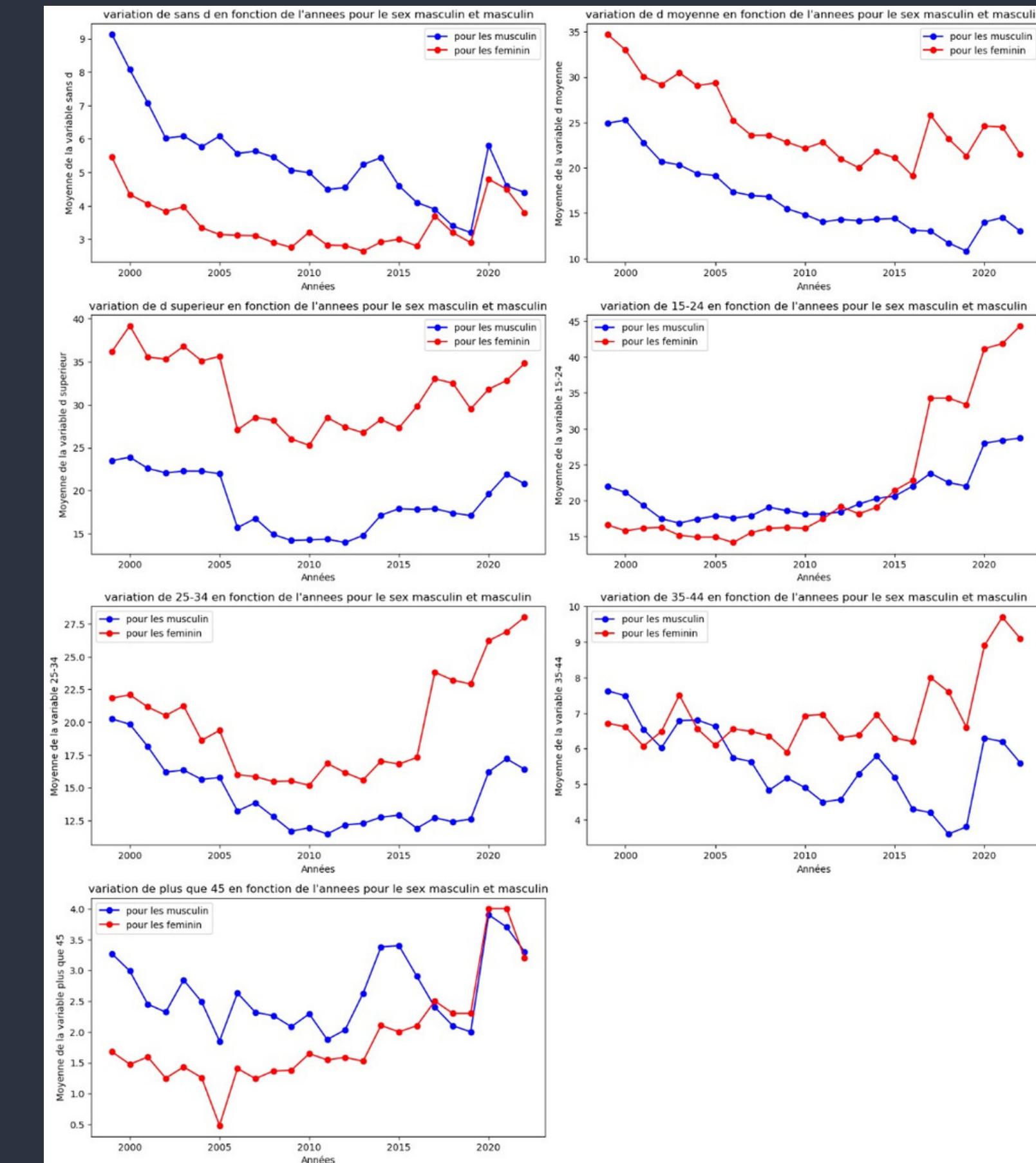
Visualisation Variation de taux de chômage par sexe au fil des années:

```
target = ['sans d','d moyenne','d supérieur','15-24','25-34','35-44','plus que 45']
plt.figure(figsize = (15, 30))
y=1
for i in target:
    moyeen_par_annees_sexe = data_final.groupby(['sex', 'Années'])[i].mean()
    par_sexe_M = moyeen_par_annees_sexe[1]
    par_sexe_F = moyeen_par_annees_sexe[-1]

    plt.subplot(4,2,y)
    plt.plot(par_sexe_M.index, par_sexe_M.values, marker='o', linestyle='--', color='b', label='pour les masculin')
    plt.plot(par_sexe_F.index, par_sexe_F.values, marker='o', linestyle='--', color='r', label='pour les feminin')
    plt.legend()
    plt.xlabel('Années')
    plt.ylabel('Moyenne de la variable {}'.format(i))
    plt.title("variation de {} en fonction de l'annees pour le sex masculin et masculin".format(i))
    y+=1
plt.tight_layout()
```

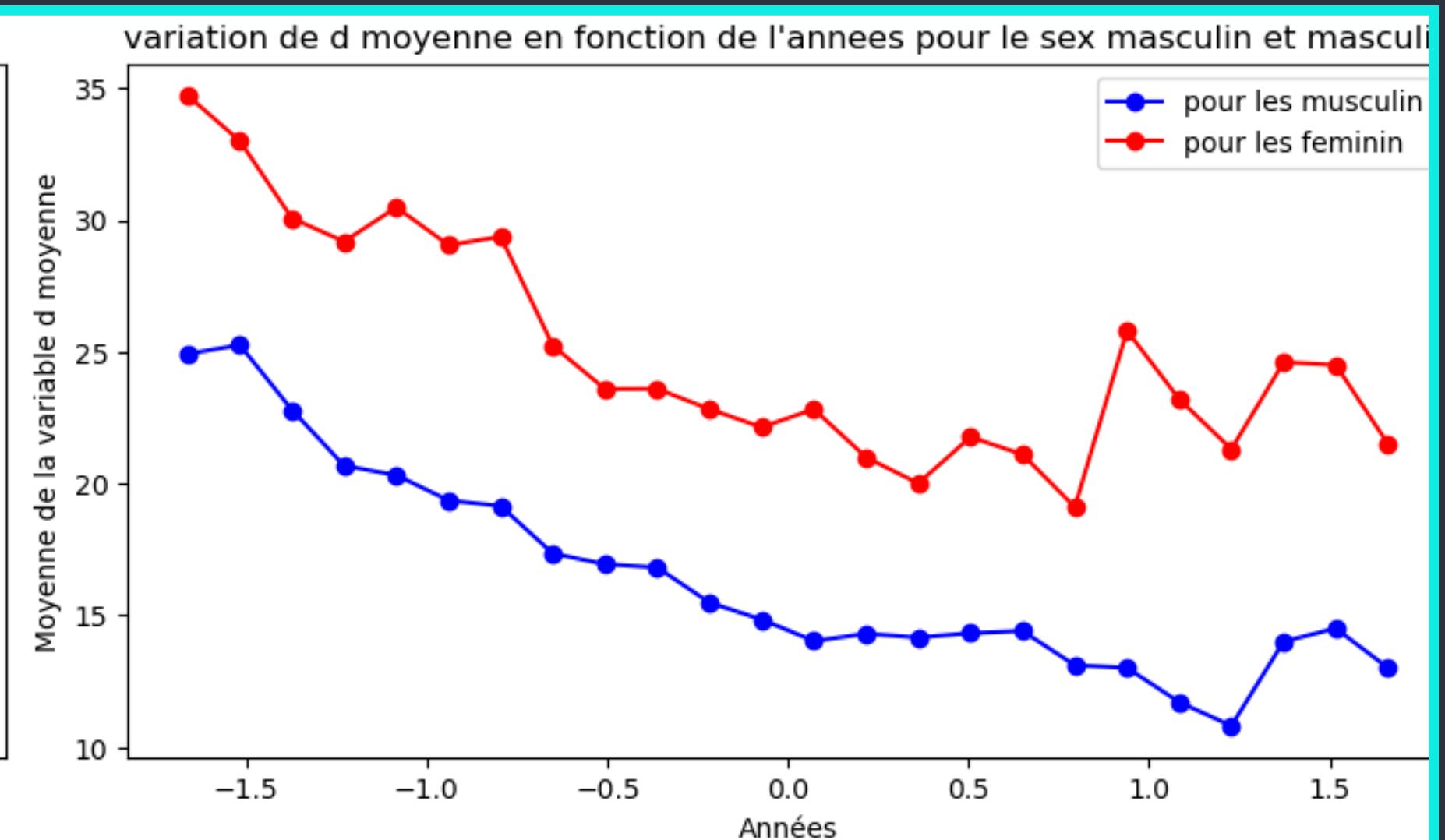
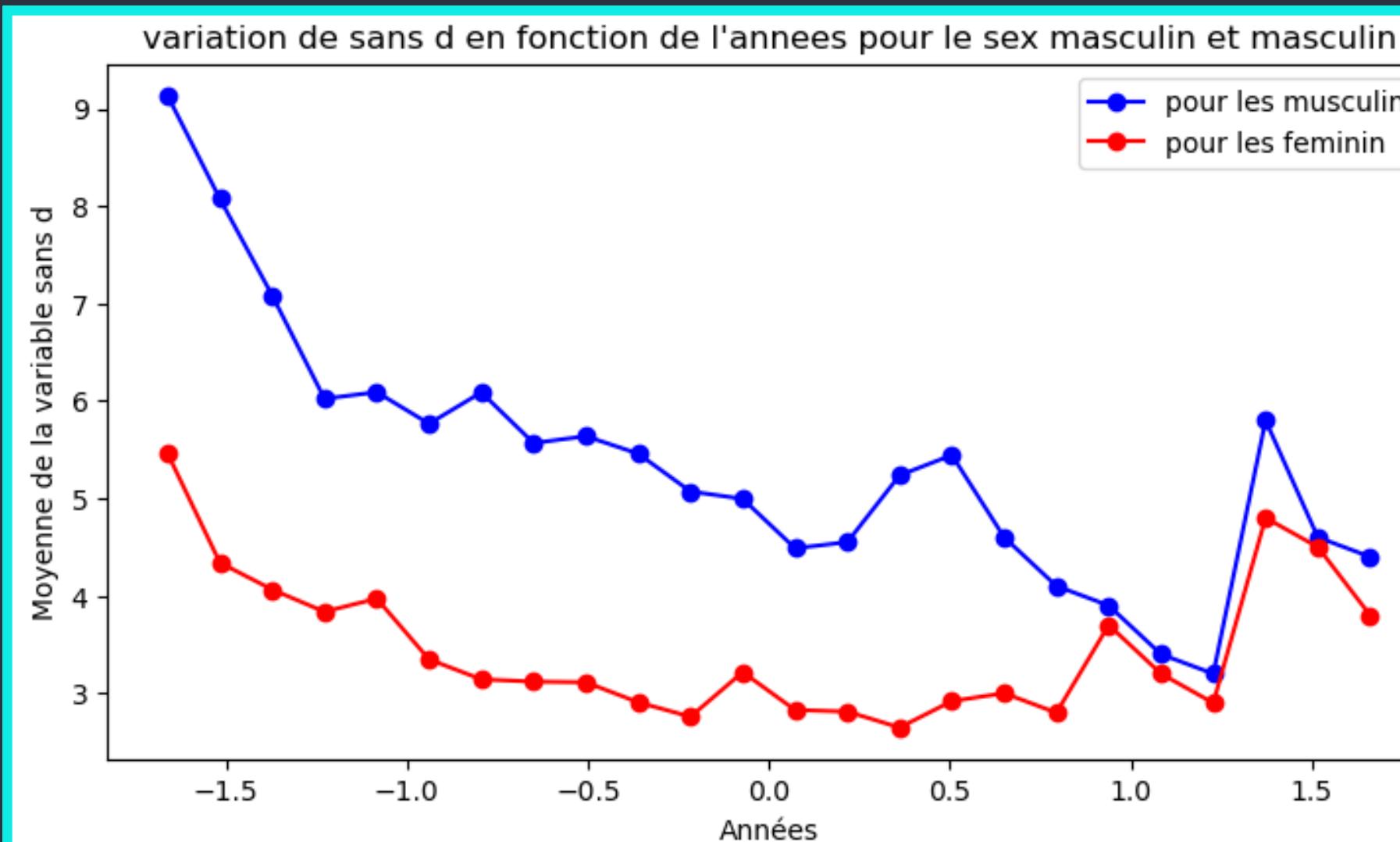
# 3. VISUALISATION DES DONNÉES

Visualisation Variation de taux  
de chômage par sexe au fil des  
années :

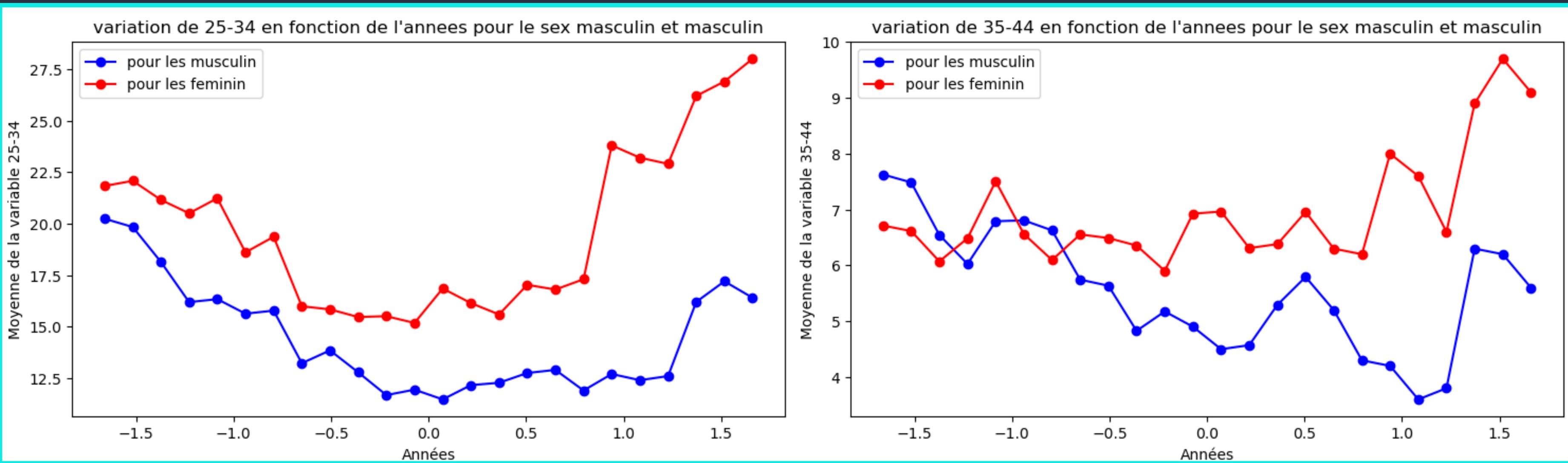


# 3. VISUALISATION DES DONNÉES

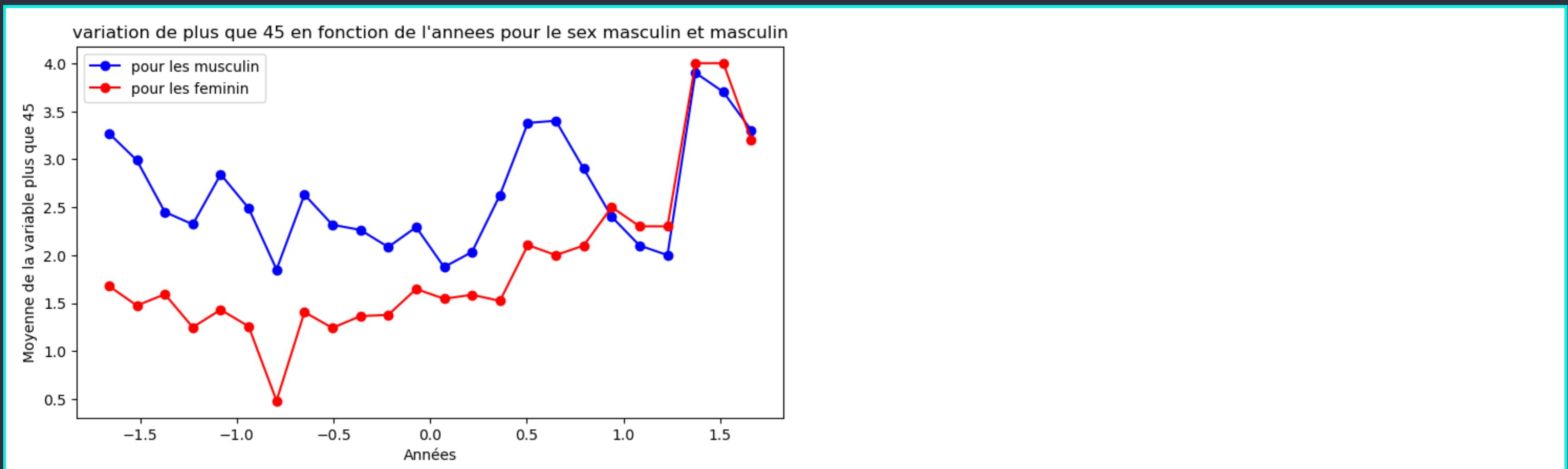
les graphes obtenus :



# 3. VISUALISATION DES DONNÉES



# 3. VISUALISATION DES DONNÉES



# 3. VISUALISATION DES DONNÉES

La visualisation de distribution des valeurs de différents variables par la création des boxplots :

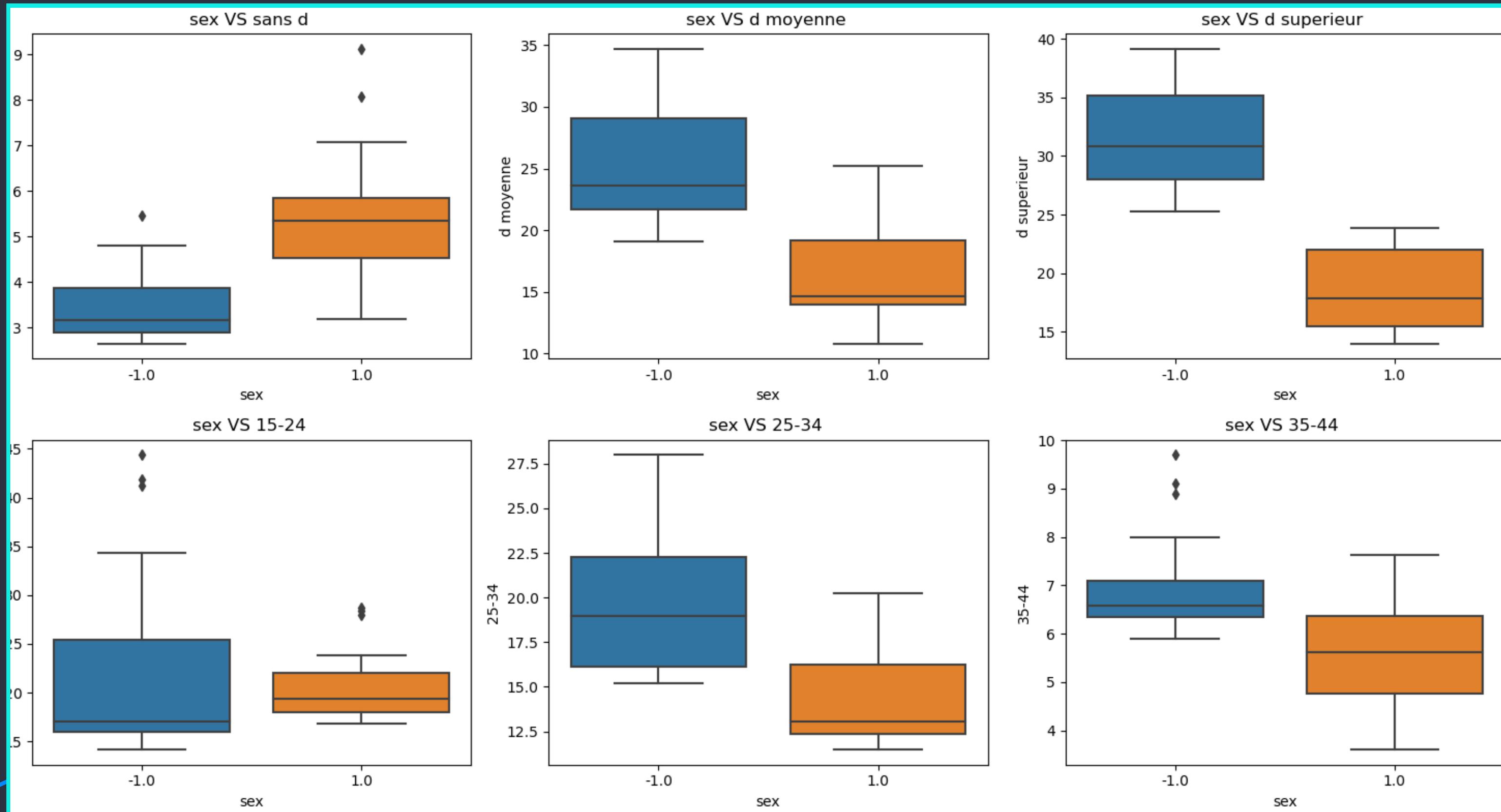
```
: target = ['sans d', 'd moyenne', 'd supérieur', '15-24', '25-34', '35-44', 'plus que 45']

fig, ax = plt.subplots(2, 3, figsize=(15, 8))

y = 0
for i in range(2):
    for j in range(3):
        if y < len(target):
            sns.boxplot(x='sex', y=target[y], data=data_final, ax=ax[i, j]).set_title('sex VS {}'.format(target[y]))
            y += 1

plt.tight_layout()
plt.show()
```

# 3. VISUALISATION DES DONNÉES



# 3. VISUALISATION DES DONNÉES

La standarisation des données pour la construction des modèles :

```
: features = ['Annes', 'sex', 'effet sanitaire', 'investissement']

scaler = StandardScaler()

data_final[features] = scaler.fit_transform(data_final[features])

: data_final.head()
```

	Annes	sans d	d moyenne	d supérieur	sex	15-24	25-34	35-44	plus que 45	effet sanitaire	investissement	effet sanitaire	investissement	effet sanitaire	investissement
0	1.661325	4.4	13.0	20.8	1.0	28.7	16.4	5.6	3.3	-0.204390	-0.672202	0.5	1.663588	0.5	1.663588
1	1.661325	3.8	21.5	34.8	-1.0	44.4	28.0	9.1	3.2	-0.204390	-0.672202	0.5	1.663588	0.5	1.663588
2	1.516862	4.6	14.5	21.9	1.0	28.4	17.2	6.2	3.7	1.328538	-0.731341	3.0	1.596519	3.0	1.596519
3	1.516862	4.5	24.5	32.8	-1.0	41.9	26.9	9.7	4.0	1.328538	-0.731341	3.0	1.596519	3.0	1.596519
4	1.372399	5.8	14.0	19.6	1.0	28.0	16.2	6.3	3.9	3.781223	-1.108249	7.0	1.169073	7.0	1.169073

# 3. VISUALISATION DES DONNÉES

La division du Dataset en test data et training data :

```
features = ['Annes', 'sex', 'effet sanitaire', 'investissement']
target = ['sans d','d moyenne','d superieur', '15-24','25-34','35-44','plus que 45']

X_train, X_test, y_train, y_test = train_test_split(data_final[features], data_final[target], test_size=0.2, random_state=5)
```

# CHOIX DE MODELE :

on a utiliser Decision Tree Regressor, Random Forest Regressor et Gradient Boosting Regressor pour predire le taux de chomage au maroc a cause des raison suivantes :

- **Problème de régression** : Les données qu'on traite sont continues ou proches les unes des autres, ce qui indique un problème de régression plutôt qu'un problème de classification. Les modèles de régression sont appropriés pour prédire des valeurs continues.
- **Non-linéarité des relations** : Il est souligné precedement dans la matrice de correlation qu'il n'y a pas de relation linéaire entre la majorité des colonnes de votre ensemble de données. Cela justifie l'utilisation de modèles de régression non linéaires, tels que les arbres de décision, les forêts aléatoires et les boosters.
- **Nature complexe et non linéaire des données** : La complexité et la non-linéarité des données sont des facteurs clés qui motivent le choix de modèles ces modeles. les forêts aléatoires et les boosters sont capables de capturer des relations complexes et de fournir des prédictions précises.
- **MultiOutputRegressor** : La cible comprenant plusieurs colonnes nécessite l'utilisation de la fonction MultiOutputRegressor. Cela permet d'adapter le modèle à plusieurs sorties simultanément, ce qui est approprié dans le cas où on essaye de prédire plusieurs variables cibles.

# PRINCIPE DE GRIDSEARCHCV :

`GridSearchCV` est une technique d'optimisation d'hyperparamètres qui permet de rechercher la meilleure combinaison d'hyperparamètres pour un modèle donné. Le principe de `GridSearchCV` est le suivant:

## 1. Définition des hyperparamètres à tester :

Vous spécifiez une grille (ou un ensemble) d'hyperparamètres que vous souhaitez optimiser. Cela peut inclure des paramètres tels que la profondeur maximale d'un arbre de décision, le nombre d'estimateurs dans un modèle d'ensemble, le taux d'apprentissage dans un algorithme de gradient boosting, etc.

## 2. Choix du modèle :

Nous sélectionnons le modèle que nous souhaitons entraîner et optimiser. Cela peut être n'importe quel modèle de scikit-learn qui expose des paramètres à ajuster.

## 3. Validation croisée :

Nous spécifiez le nombre de plis pour la validation croisée (`cv`), qui divise votre ensemble de données en plusieurs parties. Le modèle sera entraîné sur certaines parties et évalué sur d'autres. Cela aide à obtenir une évaluation plus robuste des performances du modèle.

# PRINCIPE DE GRIDSEARCHCV :

## 4. Création de la grille de recherche :

`GridSearchCV` crée toutes les combinaisons possibles des hyperparamètres que vous avez spécifiés dans la grille. Chaque combinaison est utilisée pour entraîner et évaluer le modèle.

## 5. Évaluation de la performance :

Le modèle est entraîné et évalué pour chaque combinaison d'hyperparamètres à l'aide de la validation croisée. La métrique spécifiée (par exemple, précision, F1-score, erreur quadratique moyenne, etc.) est utilisée pour évaluer la performance.

## 6. Meilleurs hyperparamètres :

Une fois que toutes les combinaisons ont été évaluées, `GridSearchCV` identifie la combinaison d'hyperparamètres qui a produit les meilleures performances selon la métrique spécifiée.

# PRINCIPE DE GRIDSEARCHCV :

## 7. Entraînement du modèle final :

- Nous pouvons utiliser les meilleurs hyperparamètres pour entraîner le modèle sur l'ensemble complet de données (ou sur un ensemble d'entraînement particulier) afin d'obtenir le modèle final. `GridSearchCV` simplifie le processus d'exploration des hyperparamètres en automatisant la recherche exhaustive, permettant ainsi de trouver la meilleure configuration d'hyperparamètres pour maximiser les performances du modèle. Cependant, cela peut être coûteux en termes de temps de calcul, surtout si la grille de recherche est grande.
- Remarque nous pouvons utiliser aussi la courbe de selectionne ou bien des boucle repititive pour obtenire des matrices carres contiens des scors pour chaque model lié par un certains hyperparamètre et on prendre la moyenne plus elevé

# GRADIENT BOOSTING REGRESSOR :

Cet exemple de code utilise la recherche par grille (`GridSearchCV`) avec la classe `MultiOutputRegressor` pour optimiser les hyperparamètres d'un modèle de régression par Gradient Boosting (`GradientBoostingRegressor`), alors on va expliquer chaque partie de ce code :

## 1. Définition des hyperparamètres à optimiser :

- `n\_estimators`: Le nombre d'arbres dans le modèle. Plus le nombre est élevé, plus le modèle est complexe.
- `learning\_rate`: Le taux d'apprentissage contrôle la contribution de chaque arbre au modèle. Une valeur plus basse nécessite un nombre plus élevé d'arbres pour atteindre la même complexité.
- `max\_depth`: La profondeur maximale de chaque arbre. Contrôle la complexité de chaque arbre.

## 2. Crédit de la grille de recherche (`param\_grid`):

- La grille de recherche spécifie toutes les combinaisons possibles des valeurs d'hyperparamètres à tester.

# GRADIENT BOOSTING REGRESSOR :

## 3. Création de l'estimateur (`GradientBoostingRegressor`):

- La classe `MultiOutputRegressor` est utilisée pour traiter plusieurs sorties simultanément. Dans cet exemple, elle est appliquée à un `GradientBoostingRegressor`.

## 4. Création de l'objet `GridSearchCV`:

- L'objet `GridSearchCV` prend l'estimateur, la grille d'hyperparamètres, la métrique de performance (`scoring`), et le nombre de plis pour la validation croisée (`cv`).

## 5. Exécution de la recherche par grille :

- `GridSearchCV` ajuste le modèle pour chaque combinaison d'hyperparamètres et évalue la performance à l'aide de la validation croisée.

## 6. Identification des meilleurs hyperparamètres :

- Une fois la recherche par grille terminée, `best\_params\_` contient les valeurs d'hyperparamètres qui ont donné les meilleurs résultats.

# GRADIENT BOOSTING REGRESSOR :

## 7. Entraînement du meilleur modèle :

- Le meilleur modèle est ensuite ajusté sur l'ensemble d'entraînement complet avec les hyperparamètres optimaux.

## 8. Prédictions et évaluation de la performance :

- Le modèle est utilisé pour faire des prédictions sur les ensembles d'entraînement (`y\_train\_pred`) et de test (`y\_test\_pred`).
- La performance du modèle est évaluée à l'aide du coefficient de détermination ( $R^2$ ) sur les ensembles d'entraînement et de test.

## 9. Affichage des résultats :

- Les résultats finaux, y compris les meilleurs hyperparamètres et les scores  $R^2$ , sont affichés.

Cet exemple illustre comment utiliser la recherche par grille pour ajuster les hyperparamètres d'un modèle de Gradient Boosting avec prise en charge de multiples sorties grâce à `MultiOutputRegressor`.

# GRADIENT BOOSTING REGRESSOR :

```
param = {  
    'estimator__n_estimators': [50, 100, 200],  
    'estimator__learning_rate': [0.01, 0.1, 0.2],  
    'estimator__max_depth': [3, 4, 5],  
}  
  
grid = GridSearchCV(MultiOutputRegressor(estimator=GradientBoostingRegressor()), param_grid=param, scoring='r2', cv=5)  
  
grid.fit(X_train, y_train)  
  
best_params = grid.best_params_  
print("Meilleurs hyperparamètres : ", best_params)  
best_model = grid.best_estimator_  
  
# Faire des prédictions avec le meilleur modèle  
y_train_pred = best_model.predict(X_train)  
y_test_pred = best_model.predict(X_test)  
  
# Évaluer la performance  
train_r2 = r2_score(y_train, y_train_pred)  
test_r2 = r2_score(y_test, y_test_pred)  
print('Train R² score:', train_r2)  
print('Test R² score:', test_r2)  
  
Meilleurs hyperparamètres : {'estimator__learning_rate': 0.01, 'estimator__max_depth': 3, 'estimator__n_estimators': 200}  
Train R² score: 0.9479313453909388  
Test R² score: 0.8452260052968742
```

# INTERPRETATION :

voici une cas d'utilisations on va predire le taux de chomage pour 2030 dans les conditions suivant: evenement mondial coup du monde alors les investissement augmenteront jusqu'à 8.66 milliard et le maroc a sortie d'une crise sanitaire depuis 2 ans avant 2030

```
In [217]: new_data={'Annes':2030,'sex':[1,-1],'effet sanitaire':2,'investissement':8.66}  
new_data=pd.DataFrame(new_data,columns=['Annes', 'sex', 'effet sanitaire', 'investissement'])  
new_data
```

```
Out[217]:
```

	Annes	sex	effet sanitaire	investissement
0	2030	1	2	8.66
1	2030	-1	2	8.66

```
In [218]: taux_chomage = best_model.predict(new_data)  
taux_chomage=pd.DataFrame(taux_chomage,columns=['sans d','d moyenne','d superieur','15-24','25-34','35-44','plus que 45'])  
taux_chomage
```

```
Out[218]:
```

	sans d	d moyenne	d superieur	15-24	25-34	35-44	plus que 45
0	5.677926	14.216952	20.372110	25.621112	16.345246	6.027122	3.412469
1	5.071961	23.086519	32.316922	40.754667	26.282883	8.455695	3.278156

```
In [219]: print("2022\n",data_final.loc[data_final['Annes']==2022,['sans d','d moyenne','d superieur','15-24','25-34','35-44','plus que 45'])  
2022  
Empty DataFrame  
Columns: [sans d, d moyenne, d superieur, 15-24, 25-34, 35-44, plus que 45]  
Index: []
```

# RANDOM FOREST REGRESSOR :

```
param = {  
    'estimator__n_estimators': [1000, 3000],  
    'estimator__max_depth': [None, 20],  
    'estimator__random_state': [42, 567]  
}  
  
grid = GridSearchCV(MultiOutputRegressor(estimator=RandomForestRegressor()), param_grid=param, scoring='r2', cv=5)  
grid.fit(X_train, y_train)  
  
best_params = grid.best_params_  
print("Meilleurs hyperparamètres : ", best_params)  
best_model = grid.best_estimator_  
  
# Faire des prédictions avec le meilleur modèle  
y_train_pred = best_model.predict(X_train)  
y_test_pred = best_model.predict(X_test)  
  
# Évaluer la performance  
train_r2 = r2_score(y_train, y_train_pred)  
test_r2 = r2_score(y_test, y_test_pred)  
print('Train R² score:', train_r2)  
print('Test R² score:', test_r2)  
  
Meilleurs hyperparamètres : {'estimator__max_depth': None, 'estimator__n_estimators': 3000, 'estimator__random_state': 42}  
Train R² score: 0.9729358598351627  
Test R² score: 0.8648293675052423
```

# INTERPRETATION :

Les résultats fournis pour RandomForestRegressor comprennent les meilleurs hyperparamètres sélectionnés par la recherche sur grille (GridSearchCV) :

- `estimator_max_depth : None` --> La profondeur maximale des arbres est réglée sur "Aucune limite", ce qui signifie que les arbres peuvent se développer jusqu'à ce que chaque feuille contienne un seul point ou qu'un autre critère d'arrêt soit atteint.
- `estimator_n_estimators : 3000` --> Le nombre d'arbres dans l'ensemble est réglé sur 3000, indiquant la taille de l'ensemble d'arbres.
- `estimator_random_state : 42` --> Le générateur de nombres aléatoires est fixé à 42 pour assurer la reproductibilité des résultats. R squared score sur le data du training 0.9729 et R squared score sur le data du test est 0.8648 ce qui montrant une excellente performance sur l'ensemble d'entraînement et une bonne capacité de généralisation sur l'ensemble de test

# DECISION TREE REGRESSOR :

```
decision_tree =DecisionTreeRegressor(random_state=5)
param_grid = {
    'max_depth': [k for k in range(1,100,10)],
    'min_samples_split': [k for k in range(2,6)],
    'min_samples_leaf': [k for k in range(1,5)]
}

grid_search = GridSearchCV(decision_tree, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

print('Les meilleurs paramètres sont:', grid_search.best_params_)

best_model = grid_search.best_estimator_
best_model.fit(X_train,y_train)
x=best_model.score(X_test,y_test)
y=best_model.score(X_train,y_train)
print('test score : ', x, 'train score', y)

Les meilleurs paramètres sont: {'max_depth': 11, 'min_samples_leaf': 1, 'min_samples_split': 4}
test score :  0.8747152427732499 train score 0.9642488228751727
```

# INTERPRETATION :

Les résultats fournis pour DecisionTreeRegressor comprennent les meilleurs hyperparamètres sélectionnés par la recherche sur grille (GridSearchCV) :

- `max_depth : 11` --> La profondeur maximale de l'arbre de décision est fixée à 11, ce qui limite la croissance de l'arbre.
- `min_samples_leaf : 1` --> Le nombre minimum d'échantillons requis pour être dans une feuille est fixé à 1, indiquant que chaque feuille peut contenir un seul échantillon.
- `min_samples_split : 4` --> Le nombre minimum d'échantillons requis pour diviser un nœud est fixé à mean square error score sur le data du training 0.96424 et sur le data du test est 0.8747 ce qui montrant une excellente performance sur l'ensemble d'entraînement et une bonne capacité de généralisation sur l'ensemble de test

# CONCLUSION:

En conclusion, ce projet de prédiction du taux de chômage au Maroc a été mené avec succès à travers des étapes rigoureuses, de la collecte de données à la validation des modèles. Les résultats obtenus démontrent la robustesse de nos modèles, qui peuvent fournir des prévisions fiables pour guider les décisions économiques. La combinaison de la visualisation des données et de l'interprétation approfondie a enrichi notre compréhension des dynamiques du marché du travail marocain. Ces outils prédictifs offrent une base solide pour des actions proactives visant à atténuer les effets du chômage et à soutenir la croissance économique.